Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2007

Quiz II

There are 11 questions and 9 pages in this quiz booklet. Answer each question according to the instructions given. You have **50 minutes** to answer the questions.

Most questions are multiple-choice questions. Next to each choice, circle the word **True** or **False**, as appropriate. A correct choice will earn positive points, a wrong choice may earn negative points, and not picking a choice will score 0. The exact number of positive and negative points for each choice in a question depends on the question and choice. The maximum score for each question is given near each question; the minimum for each question is 0. Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below AND at the bottom of each page of this booklet.

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO COMPUTERS, NO LAPTOPS, NO PDAS, ETC.

CIRCLE your recitation section number:

10:00	1. Madden/Komal		
11:00	2. Madden/Zhang	3. Katabi/Komal	10. Yip/Chachulski
12:00	4. Yip/Zhang	5. Katabi/Chachulski	
1:00	6. Ward/Shih	7. Girod/Schultz	
2:00	8. Ward/Schultz	9. Girod/Shih	

Do not write in the boxes below

1-6 (xx/36)	7-10 (xx/39)	11 (xx/25)	Total (xx/100)

Name:

I Reading Questions

1. [4 points]: A choice motivated by an *end-to-end argument* in the design of the Ethernet (as described in reading #9) is

(Circle the BEST answer)

A. Carrier detection.

B. Packet error detection.

C. Collision detection.

D. Best effort packet delivery.

E. None of the above.

2. [4 points]: Minimum packet size is constrained by maximum Ethernet (as described in reading #9) cable length because of considerations relating to

(Circle the BEST answer)

A. Carrier detection.

B. Packet error detection.

C. Collision detection.

D. Best effort packet delivery.

E. None of the above.

3. [4 points]: The ordering of responses to a Google search (reading #13) is determined solely by

(Circle the BEST answer)

A. Auction among Google clients.

B. Number of citations.

C. Page ranks.

D. Hit lists.

E. None of the above.

Name:

4. [4 points]: The LFS authors (reading #15) distinguish between ”hot” and ”cold” segments in order to
(Circle the BEST answer)

A. Control the temperature on the surface of the disk

B. Minimize garbage collection overhead

C. Optimize performance through the use of RAM caches

D. Prioritize redundant storage of most valuable data

E. None of the above

ACo, BCo, and CCo each pay their internet service provider, ICo, a femtopenny per packet for handling internet traffic. ACo's CFO, affectionately dubbed Peering Tom by his colleagues, has negotiated a deal between ACo and BCo for the free, unconstrained transfer of packets between the two companies. Routing information is exchanged between all four companies using BGP4.

5. [10 points]: After a single gateway is set up connecting ACo and BCo, ACo's bill from ICo drops drastically, while BCo's costs increase. Which of the following are possible explanations?
(Circle True or False for each choice.)

A. True / False ACo has a transit relationship with ICo, while BCo has a peering relationship.

B. True / False ACo and BCo have set MED attributes differently.

C. True / False BCo advertises its connection to ICo and elsewhere, while ACo only advertises connections within ACo itself.

D. True / False ACo uses distance-vector routing while BCo uses path-vector routing protocols.

E. True / False ACo sends more packets to BCo than BCo sends to ACo.

6. [10 points]: Which of the following describe the BGP4 connection between the gateways at ACo and BCo?
(Circle True or False for each choice.)

A. True / False It uses TCP.

B. True / False It uses iBGP rather than eBGP.

C. True / False It carries route updates.

D. True / False It carries all data packets exchanged directly between hosts at ACo and BCo.

E. True / False It carries keep-alive packets.

Name:

II File System for Dummies

Mystified by the complexity of NFS, Moon Microsystems guru Jill Boy decides to implement a simple alternative she calls File System for Dummies, or FSD. She implements FSD in two pieces:

A. An FSD server, implemented as a 2-page Python application. The server may be run by any user, and responds to FSD requests on port 11111. Each request corresponds exactly to a UNIX file system call (e.g. `read`, `write`, `open`, `close`, or `create`) and returns just the information returned by that call (status, integer file descriptor, data, etc).

B. An FSD client library, which can be linked together with various applications to substitute Jill's FSD implementations of file system calls like `open`, `read`, and `write` for their UNIX counterparts. For clarity, we refer to the FSD versions of these procedures as `fsd_open`, etc.

Jill's client library uses the standard UNIX calls to access local files, but uses names of the form

/`fsd`/*hostname*/*apath*

to access the file whose absolute path name is */apath* on the host named *hostname*. Her library procedures recognize operations involving remote files (e.g.

```
fsd_open("/fsd/csail.mit.edu/foobar", READONLY)
```

and translates them to RPC requests on port 1111 of the appropriate host, using the filename on that host (e.g.

```
RPC("/fsd/csail.mit.edu/foobar", "open", "/foobar", READONLY)).
```

The RPC call causes the corresponding UNIX command (e.g.,

```
open("/foobar", READONLY))
```

to be executed on the remote host, and the results (e.g. a file descriptor) to be returned as the result of the `RPC` call. Jill's server code catches errors in the processing of each request, and returns `ERROR` from the `RPC` call on remote errors.

Figure 1 on page 5 describes pseudocode for Version 1 of Jill's FSD client library. The `RPC` calls in the code relay simple RPC commands to the server, using *exactly-once* semantics. Note that no data caching is done either by the server or client library.

Name:

```
// Map FSD handles to host names, remote handles:
string handle_to_host_table[1000];     // initialized to UNUSED
int handle_to_rhandle_table[1000];     // handle translation table

procedure fsd_open(string name, int mode)
  integer handle = find_unused_handle();

  if name begins with "/fsd/" then {
     host = extract_host_name(name);
     filename = extract_remote_filename(name);

     // returns file handle on remote server, or ERROR
     rhandle = RPC(host, "open", filename, mode);
     }
  else {
     host = "";
     rhandle = open(name, mode);
     }

  if rhandle == ERROR then return ERROR;

  handle_to_rhandle_table[handle] = rhandle;
  handle_to_host_table[handle] = host;
  return handle;

procedure fsd_read(int handle, string buffer, int nbytes)

   host = handle_to_host_table[handle];
   rhandle = handle_to_rhandle_table[handle];

   if host == "" then return read(rhandle, buffer, nbytes);

   // The following call sets "result" to the return value from
   // the read(...) on the remote host, and copies data read into buffer:
   result, buffer = RPC(host, "read", rhandle, nbytes);

   return result;

procedure fsd_close(int handle)
   host = handle_to_host_table[handle];
   rhandle= handle_to_rhandle_table[handle];
   handle_to_rhandle_table[handle] = UNUSED;

   if host = "" then return close(rhandle);
   else return RPC(host, "close", rhandle);
```

Figure 1: Pseudocode for FSD client library, Version 1

Name:

7. [4 points]: What does the above code indicate via an empty string (" ") in an entry of handle_to_host_table?
(Circle the BEST answer)

A. An unused entry of the table.

B. An open file on the client host machine.

C. An end-of-file condition on an open file.

D. An error condition.

E. None of the above.

Mini Malcode, an intern assigned to Jill, proposes that the above code be simplified by eliminating the `handle_to_rhandle_table` and simply returning the untranslated handles returned by `open` on the remote or local machines. Mini implements her simplified client library, making appropriate changes to each `fsd_` call, and tries it on several test programs.

8. [12 points]: Which of the following test programs will continue to work after Mini's simplification? Mark TRUE if the program will work, or FALSE if it is likely to encounter bugs.
(Circle True or False for each choice.)

A. True / False A program that reads a single, local file.

B. True / False A program that reads a single remote file.

C. True / False A program that reads and writes many local files.

D. True / False A program that reads and writes several files from a single remote FSD server.

E. True / False A program that reads many files from different remote FSD servers.

F. True / False A program that reads several local files as well as several files from a single remote FSD server.

Name:

9. [16 points]: Jill rejects Mini's suggestions, insisting on Version 1 code shown above. She is asked by marketing for a comparison between FSD and NFS. Complete the following table comparing NFS to FSD, by circling yes or no under each of NFS and FSD for each statement:

Statement	**NFS**		**FSD**	
remote handles include inode numbers	Yes	No	Yes	No
read, write calls are idempotent	Yes	No	Yes	No
can continue reading an open file after deletion (e.g. by program on remote FSD host)	Yes	No	Yes	No
requires mounting remote file systems prior to use	Yes	No	Yes	No

Convinced by Moon's networking experts that a much simpler RPC package promising *at-least-once* rather than *exactly-once* semantics will save money, Jill substitutes the simpler RPC framework and tries it out. Although the new (Version 2) FSD works most of the time, Jill finds that an `fsd_read` sometimes returns the wrong data; she asks you to help. You trace the problem to multiple executions of a single RPC request by the server, and are considering various suggestions for eliminating the bug while continuing to use the new RPC package. Among the additions you are considering are

- A response cache on the client, sufficient to detect identical requests and returning a cached result for duplicates without re-sending the request to the server;
- A response cache on the server, sufficient to detect identical requests and returning a cached result for duplicates without re-executing them;
- A monotonically increasing *sequence number* (nonce) added to each RPC request, making otherwise identical requests distinct.

10. [7 points]: Which of the following changes would you suggest to address the problem introduced by the *at-least-once* RPC semantics?

(Circle the BEST answer)

A. Response cache on client.

B. Response cache on server.

C. Sequence numbers in RPC requests.

D. Response cache on client AND sequence numbers.

E. Response cache on server AND sequence numbers.

F. Response caches on both client and server.

Name:

III RaidCo

RaidCo is a company that makes pin-compatible hard disk replacements using tiny, chip-sized hard disks ("microdrives") that have become available cheaply. Each RaidCo product behaves like a hard disk, supporting the operations

```
error = read(nblocks, starting_block_number, buffer_address)
error = write(nblocks, starting_block_number, buffer_address)
```

to read or write an integral number of consecutive blocks from or to the disk array. Each operation returns a status word indicating whether an error has occurred.

RaidCo builds each of its disk products using twelve tiny, identical microdrives configured as a RAID system. A team of ace 6.033 students designed RaidCo's system, and they did a flawless job of implementing six different RaidCo disk models. Each model uses identical hardware (including a processor and the twelve microdrives), but the models use different levels of RAID in their implementation and offer varying block sizes and performance characteristics to the customer. Note that the RAID systems' block sizes are not necessarily the same as the sector size of the component microdrives.

The models are

- **R0** sector-level striping across all twelve microdrives, no redundancy/error correction
- **R1** six pairs of two mirrored microdrives, no striping
- **R2** 12-microdrive RAID 2 (bit-level striping, error detection, and error correction); microdrive's internal sector-level error detection is disabled.
- **R3** 12-microdrive RAID 3 (sector-level striping and error correction)
- **R4** 12-microdrive RAID 4 (no striping, dedicated parity disk)
- **R5** 12-microdrive RAID 5 (no striping, distributed parity)

The microdrives each conform to the same read/write API sketched above, each microdrive providing 100K sectors of 1K bytes each, and offering a constant 10 ms seek time and a read/write bandwidth of 100 megabytes/second; thus the entire 100MB of data on a microdrive can be fetched using a single read operation in one second. The RaidCo products do no caching or buffering: each read or write involves actual data transfer to or from the involved microdrives. Since the microdrives have constant seek time, the RaidCo products do not use RAID optimizations for seek time.

As good as the 6.033 students were at programming, they unfortunately left the documentation unfinished. Your job is to complete the following table, showing certain specifications for each model drive - i.e., the size and performance parameters of the API supported by each RAID system. Entries assume error-free operation, and ignore transfer times that are small compared to seek times encountered.

Name:

11. [25 points]: Complete the following table:

	R0	**R1**	**R2**	**R3**	**R4**	**R5**
Block Size (KB) exposed to `read`/`write`	1KB	1KB	_____	11KB	_____	1KB
# of Blocks (K)	_____	_____	_____	_____	_____	1100K
Max Time for a single 100MB read (sec)	1/12 s	_____	_____	_____	1 s	1 s
Time for 1-block write (ms)	10ms	10ms	10ms	_____	_____	20ms
typical # microdrives involved in 1-block read	1	_____	_____	_____	_____	1
typical # microdrives involved in 2-block read	_____	2	_____	_____	1	1
typical # microdrives involved in 2-block write	_____	2	_____	_____	_____	_____

End of Quiz II

Name: